メタモデルに基づいた図式的モデル記述と検証のための支援環境

4K－10

金村 星吉　　上田 賀一

茨城大学工学部情報工学科

1　はじめに

ソフトウェア開発過程において，モデルを作成することで複雑性が減少し，問題領域の全体構造の把握が容易にできる．モデルの表記法としては言語表現や図式表現などが存在するが，ソフトウェア開発においては図式表現，特にグラフ表現が多く用いられている．これは状態遷移図やデータ・フロー図にあたる．本研究ではこのようなモデルを統一的に，かつ効率良く構築，実行するための環境の設計およびそれらの実装をした．このような環境のもとでモデルを構築することにより，ソフトウェア開発段階の複雑性，問題領域の曖昧性が減少する．

2　対象とするモデル

状態遷移図やデータ・フロー図などのモデルの意味的側面は，Entity と Relationship の２種類の要素で構成されていると考えられる．いわゆる，ER モデルと呼ばれるものである．しかし，従来の ER モデルではモデルは書けるが振舞いや属性は存在しない．

本研究ではシミュレーション可能なモデル構築を目標としているので，オブジェクト指向概念を取り入れ Entity と Relationship に振舞いと属性を持たせた OER(Objective Entity Relationship) モデルを用いることにした．これらの考えとともに Token オブジェクトという概念も使う．Token も Entity と Relationship と同じく振舞と属性を持たせている．役割としては，Entity と Entity の間の Relationship を流れていくメッセージオブジェクトとして使われる．これら三つのいずれかのオブジェクトに属するものの組合せで，対象となるモデルを作成する．Entity, Relationship, Token はそれぞれメタモデルからのメタコンポーネントである．

3　モデルの検証

モデル開発において構築されたモデルの正当性を検証するための何らかの基準が必要である．ここでは，それらの基準を与える枠組として，モデルに対して制約を適用することにする．更にその制約は，ターゲットとするモデルによって変化するものである．よって，そのターゲットに応じてカスタマイズ可能でなければならない．

本研究では制約を系統立てて記述するために，意味的観点から次の三つに分類した．

- 構造の制約
- クラスの制約
- 操作の制約

3.1　構造の制約

構造の制約とは，モデルがどのような構造になっているかを規定するものである．今回，モデルの構造は静的なものを扱うこととしているので構造の制約を調べるタイミングは以下の時が考えられる．

- Entity と Entity を関連づける時
- モデルを実行する時

前者の場合は実現が容易であるが，二つ以上の並びを規定するようなものはチェックできない．

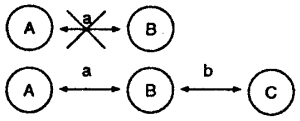

図. 構造の制約

例えばこの図の示す制約とは A と B を a で関連づけるのは不可能であるが，A と B が a で関連づけられ，かつ B と C が b で関連づけられた時のみ，制約を満たすと考える．この様な任意の長さの制約が規定されている状況において，モデルがそれらを満たしているかどうかを調べるには後者の場合が比較的容易である．よって，本研究では後者を選択することにした．

今回，記述の容易性や理解の容易性を第一とし，並びのみをサポートすることにした．Relationhip の記述に関しては方向を考慮した記述もできるようにする．

3.2　クラスの制約

クラスの制約とはそのクラスに属する Entity が常に満たさなければならない制約である．すなわち，その Entity,Relationship の属性などに課す制約である．

Supporting environment for metamodel based visual model description and verification

Seikichi Kanemura, Yoshikazu Ueda
Department of Computer and Information Sciences, Faculty of Engineering, Ibaraki University

この制約を適用するのはオブジェクトの内部に何らかの操作を加えようとした時である．よって，この制約はシミュレーション実行時に判定される．

クラスの制約を満たさない時，ユーザの定義したアクション関数を実行する．又，アクションを省略した時にはエラーアクションを起動する．

3.3 操作の制約

操作の制約とはオブジェクトに対して，ある操作を施す前，施した後にそれぞれ満たしていなければならない条件である．よって，事前制約と事後制約の二種類が存在する．クラスの制約は常に満たしているべき制約であるが，この操作の制約は操作を施す前，施した後にそれぞれ適用される制約であって，必ずしも常に満たされる必要のあるものではない．

これもシミュレーション実行時に判定されるものである．操作の制約を満たさない時，クラスの制約同様，アクションを実行する．

4 モデル開発支援環境 MDSE

モデル記述とモデル実行の間をシームレスに行なうための開発支援環境 MDSE(Model Development Supporting Environment) を与える．

MDSE では，二次元的広がりを持つモデルを効率的に開発，検証するために視覚的なユーザインターフェースと実行をサポートする．

4.1 MDSE が扱えるオブジェクトの種類

MDSE 上では先に述べた Entity, Relationship, Token の三つを扱える．このどれかに属するオブジェクトからインスタンスを生成してそれらの組合せでモデルを構築していく．ここでのオブジェクトとはモデルのメタコンポーネントを指す．

4.2 MDSE 上でのモデル編集

MDSE 上で Entity と Relationship のインスタンスの生成と削除を行ないながら，それらインスタンスを使ってモデルを構築していく．編集を終えたモデルに関してはすぐに実行に移してみて，その振舞を検証する．当然，振舞が異常な時には実行を中断して，すぐに編集し直すことも可能である．

以下に MDSE 上で扱える編集機能を列挙する．

1. インスタンスの生成と削除
2. 生成したインスタンスを用いたモデルの編集
3. モデルのセーブ・ロード

4.3 MDSE 上でのモデル実行

MDSE 上で編集したモデルは完成した時点で，すぐに実行に移すことができる．モデル実行中にはメッセージのやりとりを視覚的に表現してくれる．当然，制約を解消できなかったり，ユーザが実行の停止を指定した時には直ちに停止して，編集が行なえる．また，モデル実行に関してはモデルのデバッガの機能も兼ね備えている．デバッガとして使える主な機能を次に列挙する．

1. 制約 (構造，クラス，操作) のチェック
2. 実行前，実行中の属性値の確認・変更
3. 実行時の特定経路の通過頻度のチェック
4. 実行の一時中断と再実行
5. 一時中断後の流れの強制的変更
6. ブレイクポイントの設定・解除
7. モデルの実行トレース

4.4 MDSE の扱えるモデルの種類

MDSE 上で扱えるモデルは多岐に渡っている．従来のモデルシミュレータでは実行中に構成が変化しないモデルだけをサポートしたものが大半であった．構成が変化しないモデルの代表的な例として待ち行列，ペトリネット，論理回路などのモデルが挙げられる．しかし，モデルの構成が変化するモデルの実行をサポートしたものは少ない．

今回開発した MDSE では構成が変化するモデルもサポートしている．モデル実行中にも MDSE がインスタンスの生成，削除，そして属性の変化を監視しており，その変化をリアルタイムに表示してくれる．このような機構のおかげで Token も Entity と Relationship と同じオブジェクトとして統一的に MDSE 上で扱えるようになった．

これにより従来では扱いにくかったモデルも扱えるようになった．

【参考文献】

(1) 大野賢二　清水俊吾　上田賀一: メタモデルに基づくモデル記述とシミュレーション, 情報処理学会第44回全国大会 (5)4J-8,pp225-226

(2) 阿部倫之　吉田正弘　中川秀敏: オブジェクト指向プロトタイピングのための視覚的支援環境, 情報処理学会研究報告 Vol 94,No 55,pp 137～144(1994.7)